DEEPPANORF: DEEP PRIOR FOR NEURAL 3D RECONSTRUCTION FROM SPARSE PANORAMAS

A Thesis

presented to

the Faculty of California Polytechnic State University,

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Edward Du

June 2025

COMMITTEE MEMBERSHIP

TITLE: DeepPanoRF: Deep Prior for Neural 3D Reconstruction from Sparse Panoramas

AUTHOR: Edward Du

DATE SUBMITTED: June 2025

COMMITTEE CHAIR: Jonathan Ventura, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Alex Dekhtyar, Ph.D.
Professor of Computer Science

COMMITTEE MEMBER: Lubomir Stanchev, Ph.D.
Professor of Computer Science

ABSTRACT

DeepPanoRF: Deep Prior for Neural 3D Reconstruction from Sparse Panoramas

Edward Du

Advances in neural field representations have led to a significant improvement in view synthesis quality. However, many current novel view synthesis methods rely on a dense set of input views, which can be impractical and inefficient in real-world applications. We propose DeepPanoRF, a novel method for 360° scene reconstruction from a sparse set of input equirectangular panoramas. Built upon K-Planes, a radiance field representation that encodes explicit features on orthogonal feature planes, our method does not directly learn feature grids. Instead, we parameterize the feature grids to enable sparse view reconstruction without pretraining or additional regularization. We implement a custom U-Net architecture to take advantage of the encoder-decoder network and the skip connections. We evaluate our method's effectiveness on the Habitat-Matterport 3D (HM3D) dataset, which consists of diverse, high-quality indoor environments. Our results demonstrate that DeepPanoRF outperforms K-Planes in both reconstruction quality and structural coherence when dealing with sparse input views.

ACKNOWLEDGMENTS

Thanks to:

- My parents, for their unwavering support throughout my academic career, and for giving me the freedom to pursue my career path.
- Dr. Jonathan Ventura, for introducing me to the world of computer vision, allowing me to experience research at a high level, and for the invaluable mentorship.
- Dr. Alex Dekhtyar, for helping me develop a strong foundation in data science.
- My friends, for their constant support throughout graduate school and for sharing in both the challenges and triumphs of this experience.
- Nam Nguyen, for helping me refine ideas for this research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Virtual Reality (VR) has emerged recently as a transformative technology, enabling immersive experiences in a wide range of applications, ranging from entertainment and gaming to education. The concept of VR dates back to the mid-20th century, with early experiments like Morton Heilig's Sensoram in 1962. Another push for VR was made by Nintendo in 1995, with their release of the Virtual Boy: marketed as a console capable of displaying stereoscopic 3D graphics. But, at the time, the technology was not consumer-ready. Now, companies like Meta and Apple are spearheading the development of consumer-grade VR.

A key component of the immersive experience that VR provides is 360° panoramic images. These allow users to look around a scene in all directions by filling their field of view. However, a main point of VR is the ability is to move around within a scene, which is unable to be provided for with just a single image. The idea of generating novel, unseen views from input viewpoints is called novel view synthesis. Novel view synthesis allows for a more seamless transition between viewpoints, allowing users to move within a scene. Breakthroughs in neural field representations of scenes, such as Neural Radiance Fields (NeRFs) [20], and hybrid (implicit-explicit) representations of 3D geometry, like tri-planes [3], have led to higher-fidelity view synthesis and novel optimization methods. However, these methods rely on a dense set of input views and perform poorly when only having a sparse set of inputs. In real-world applications, capturing a dense set of high-quality images is not always practical.

A recent method called ZeroRF [28] focuses on reconstruction with sparse views. They integrate a Deep Image Prior: the idea is that a generator network's structure can produce a prior [31], into a factorized NeRF representation. ZeroRF parameterizes a 4D voxel grid with per-voxel multi-channel features from TensoRF [5] with a convolutional neural network, which enables them to perform sparse view reconstruction.

We propose a method, DeepPanoRF, that can create 360° scene reconstructions from a sparse input set of just three viewpoint panoramas. We take influence from ZeroRF's method of parameterizing feature grids and K-Planes [11], which stores explicit features on three orthogonal feature planes (k-planes field grid). Instead of directly optimizing these feature grids, we use a U-Net, without prior model pre-training, to parameterize the feature grids. The intuition is that the U-Net model architecture can generalize better for 360° scenes. Our contributions are as follows:

1. We create a pipeline that reconstructs 360° panoramas using scenes from the HM3D dataset.
2. Our method, on average, outperforms K-Planes with three input viewpoints.
3. We create a U-Net that parameterizes k-plane and density feature grids.

To evaluate our method, we use scenes from the Habitat-Matterport 3D (HM3D) dataset [22]. We compare our method to K-Planes on the same scenes, using the image-similarity metrics PSNR, SSIM, and LPIPS.

Chapter 2

RELATED WORKS & BACKGROUND

DeepPanoRF is primarily built on two concepts. We utilize the base trainer of a variant of K-Planes [11] accelerated with NerfAcc [18] for accelerated ray sampling and rendering. Second, we utilize a Deep Image Prior [31], in the form of a U-Net [24] architecture, pretrained with ResNet101[13].

2.1 360° Panorama

A 360° panorama, or an equirectangular panorama, is a type of image that captures the full spherical view of a scene, both horizontally and vertically. Unlike standard perspective images, equirectangular panorama images contain information about the full surroundings, making them useful to 3D reconstruction and view synthesis for understanding scene geometry.

Figure 2.1: *Equirectangular Panorama*

2.2 Novel View Synthesis

DeepPanoRF is a method for novel view synthesis. Novel view synthesis (NVS) is the problem of generating previously unseen views from a set of input images. Each input image has a specific camera pose, which is the position and orientation of a camera in a coordinate system.

Early methods, such as plenoptic modeling [19] and light field sample interpolation [17] required a large number of densely sampled views. Two types of rendering methods have been the predominant methods in NVS: mesh-based and volume-based rendering. Mesh-based methods [9, 12] represent scenes as a polygonal mesh and provide renderings using camera transformation and projection. Meshes are efficient but struggle with rendering complex scenes because they rely on an explicit surface reconstruction. Volume-based rendering methods [27, 21, 20] model scenes as a volume. Images are rendered by sampling rays and integrating the optical properties (color, density). Volume-based methods have higher rendering quality but are more computationally expensive because they require sampling along many points along the entire ray, instead of just the surface.

Recent advancements in deep learning have led to the adoption of neural networks in NVS. These methods [32, 29] use various deep learning architectures to learn 3D scene representation with only 2D supervision, or achieve novel view synthesis from a single image.

Modern methods combine many deep learning techniques to perform NVS. MVSplat360 [7] is a method for sparse 360° view synthesis that builds upon MVSplat [6], a feed-forward 3D Gaussian Splatting reconstruction model. They use a video diffusion model, Stable Video Diffusion [2], to refine the visual appearance of the outputs

of the modified MVSplat. Yet these methods carry a high computation overhead and require the training of several complex models.

2.3 Neural Radiance Fields

Neural Radiance Fields (NeRFs) [20] have recently had a surge in popularity. In the seminal paper "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis", Mildenhall et al. represents 3D scenes as a continuous volumetric function parameterized by a fully connected neural network. Fig. 2.2 displays an overview of the NeRF process. First, 5D coordinates ((x, y, z) spatial location and (θ, ϕ) viewing direction) are sampled along camera rays. Next, those coordinates are sent to an MLP to produce a color and volume density. A differentiable volumetric rendering function is then used to compile these values into an image. Finally, the scene representation can be optimized by minimizing the residual of the total squared loss between rendered and true pixel colors.

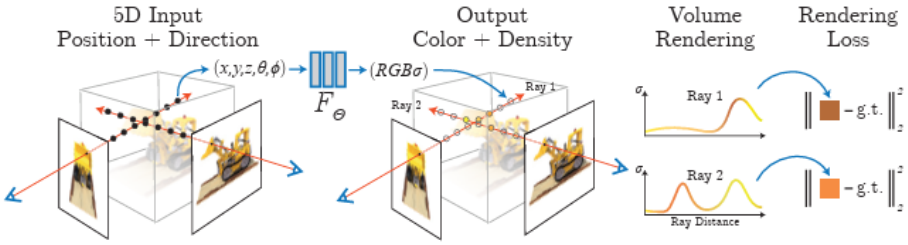


Figure 2.2: *Overview of the NeRF method. Reproduced from [20].*

However, NeRF requires many images captured from different viewpoints to learn the 3D representation of a scene. Furthermore, NeRF is computationally expensive and takes a long time to train and render.

A recent method called PanoGRF: Generalizable Spherical Radiance Fields for Wide-baseline Panoramas extends nerf to 360° views [8]. PanoGRF uses a generalizable spherical radiance field (Spherical NeRF) and a mono-guided spherical depth estima-

tor to perform NVS on panoramas. However, PanoGRF is a NeRF-based method, and has similar drawbacks.

2.4 Tri-plane Structure

Chan et al.'s 2021 paper introduces a tri-plane based 3D GAN framework and compares explicit vs implicit representations of 3D geometry [3]. Fig. 2.3 displays three different scene representations. NeRF, which is a neural implicit representation, is slow to query because it uses fully-connected layers with a positional encoding to represent scenes. Voxel grids are fast to query (as an explicit representation) but scale poorly with resolution or complex scenes. Tri-planes, which are a hybrid method, are memory-efficient and also fast to query.

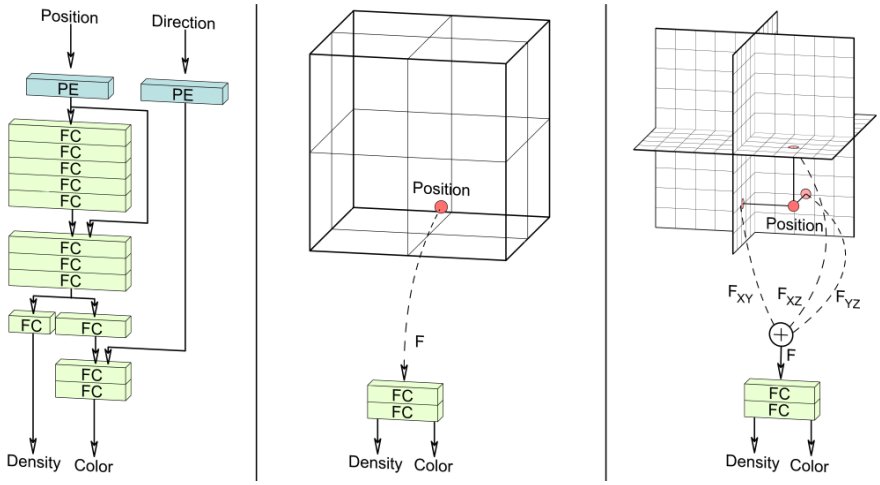


Figure 2.3: *Different representations of scenes: NeRF (left), Voxels (middle), and Tri-planes (right). Reproduced from [3].*

In Chan et al.'s tri-plane representation, explicit features are stored along three orthogonal feature planes, aligned with the 3D Cartesian coordinate system. A 3D position (x, y, z) is projected onto the three feature planes, and feature vectors (F_{xy}, F_{xz}, F_{yz}) are obtained with bilinear interpolation. The three vectors are summed

up, and a decoder network processes the resulting vector to get color and density. Finally, the RGB image is obtained with neural volume rendering.

2.5 K-Planes

Whereas a tri-plane has three feature dimensions, K-Planes is a representation that has an arbitrary number of feature dimensions. Fridovich-Keil et al.'s 2023 paper explored the use of planar representations for dynamic scenes (4D dynamic videos) [11]. While our method trains and tests on static 360° panoramic images, Fridovich-Keil et al. improves upon the tri-plane method [3]. Instead of the summation of vectors, k-planes multiplies feature vectors with the Hadamard product (elementwise multiplication). Fig. 2.4 illustrates that combining feature vectors by multiplication can produce more accurate, localized signals. This improves the rendering capabilities of linear decoders and MLP decoders.

DeepPanoRF builds on K-Planes, but instead of directly optimizing the feature grids, we parameterize them with a deep neural network.

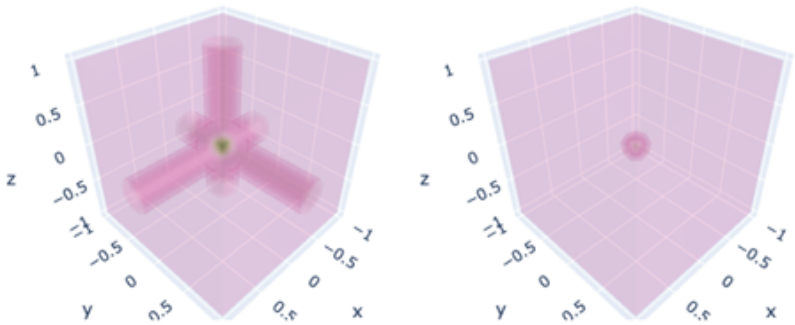


Figure 2.4: *Elementwise summation (left) compared to Hadamard product (right). Reproduced from [11].*

2.6 Deep Image Prior

Convolutional neural networks (CNNs) have been successful because of their ability to learn realistic image priors from a large set of training images. Deep Image Prior [31], however, shows that just the structure of a CNN can act as an effective image prior. When fitting a randomly initialized CNN, the network initially captures the clean structure before overfitting to noise and can avoid any potential bias towards training data.

2.7 ZeroRF

ZeroRF integrates a Deep Image Prior into a factorized NeRF representation [28]. Instead of optimizing grids from TensoRF, the authors parameterize the feature grids with a randomly initialized generator. They show that this enables sparse view reconstruction without any pretraining. We use a similar approach to parameterize our feature grids, taken from K-Planes, with a U-Net.

2.8 U-Net

A U-Net is a type of convolutional neural network designed for biomedical image segmentation[24] but has since been adopted for many other computer vision tasks because of the encoder-decoder architecture and the use of skip connections. Examples include diffusion models, generative adversarial networks, and image denoising. The architecture of the original U-Net is shown in Fig. 2.5. The shape of the network resembles a 'U', hence the name 'U-Net'.

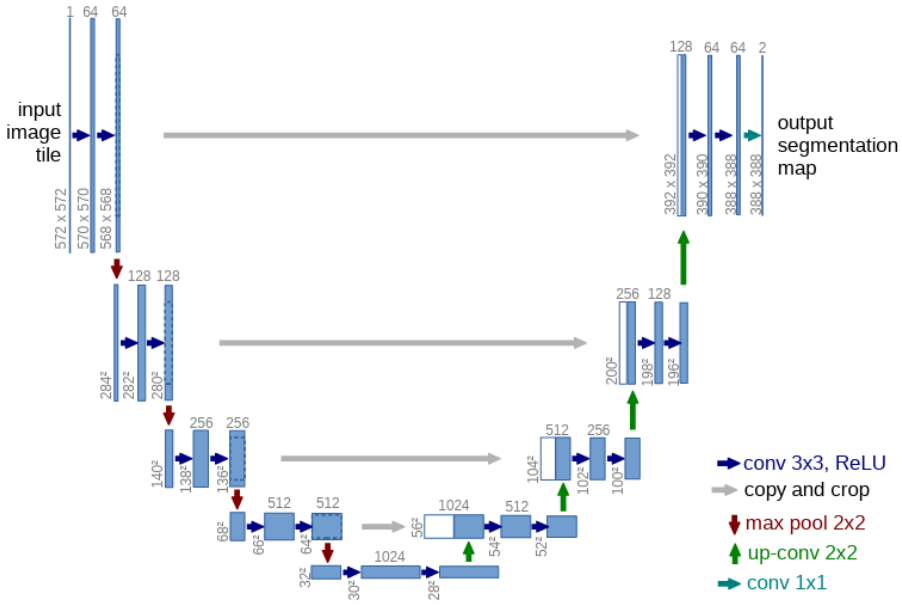


Figure 2.5: *U-Net architecture. Blue boxes correspond to a multi-channel feature map. The number of channels is labeled above the box. The input size is labeled at the bottom left of the box. White boxes represent feature maps. The arrows denote the operations. Reproduced from [24].*

In the encoder, or contracting path, the network downsamples an input while doubling the number of feature channels and extracts the intermediate features. In the decoder, or expanding path, the network upsamples an input and halves the number of feature channels. Saved copies of the encoders features, via skip connections, are concatenated onto the decoder's features. Skip connections link the contracting and expanding

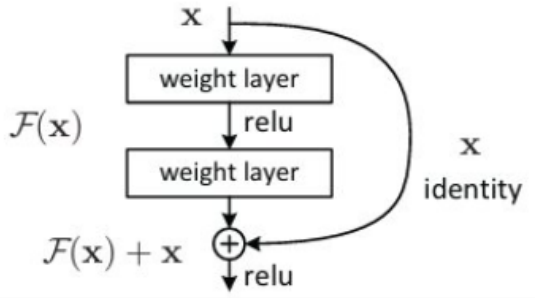


Figure 2.6: *A "shortcut connection" from ResNet [13] that performs identity mapping. Outputs are added to the outputs of stacked layers.*

paths and help retain spatial information. In DeepPanoRF, we utilize ResNet101, a 101-layer variant of ResNet, as the backbone for our U-Net architecture.

The main idea of our method is that a U-Net architecture allows for better scene reconstruction with fewer views. The idea that randomly-initialized deep neural networks (generators) enable sparse view 360° reconstruction is not new [28]. Neural networks have more resistance to noise and artifacts in data [3, 14], and deep networks can capture features prior to any learning [28].

2.8.1 ResNet Backbone

ResNet (Residual Network) [13] is a deep convolutional neural network architecture developed to address the vanishing/exploding gradient problem, a common issue in deep learning where gradients during backpropagation become zero or extremely large. ResNet uses skip connections to connect activations of a previous layer to further layers, skipping intermediate layers. The group of layers between the skip connection is called a residual block. A ResNet is created by stacking these residual blocks together. These residual blocks help mitigate the vanishing/exploding gradient problem by ensuring that instead of learning the entire output mapping, each layer only learns the "residual"—the difference needed to transform the input into the desired output. This simple formula ensures that earlier layers maintain a direct connection to the final output. As a result, gradients flow more smoothly backward through the network, greatly reducing the chance they will vanish or explode.

Chapter 3

IMPLEMENTATION

In this chapter, we detail the implementation of DeepPanoRF. Our objective is to use a single U-Net to jointly obtain both the k-planes field grids and density field grids for view synthesis. The k-planes field grid encodes the scene's appearance at different depths, and the density field grid represents the scene's spatial distribution of matter. A pipeline of our method is illustrated in Fig. 3.1.

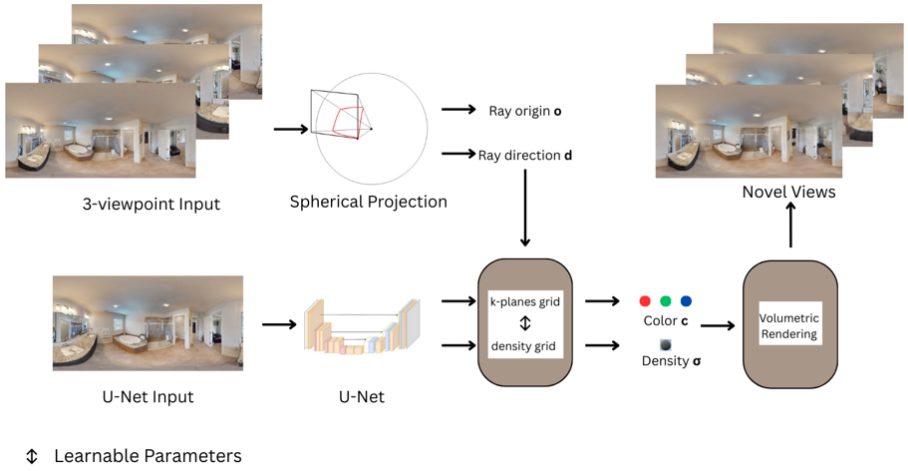


Figure 3.1: *Our DeepPanoRF pipeline, which turns sparse (3) views of a scene into novel equirectangular panorama views.*

3.1 Input

Our pipeline takes as input a set of 360° equirectangular images. We experiment with images that have resolutions of $1,024 \times 512$.

3.2 Coordinate Projection

Since our inputs are 360° equirectangular panoramas, we use a spherical projection instead of a pinhole projection. The purpose of this projection is to map each (x, y) pixel in the panorama to camera directions in unit sphere coordinates, which are used to obtain the ray origins and ray directions. We randomly sample pixels, then we convert the (x, y) image coordinates into angles.

$$\theta = 2\pi \left(\frac{x}{\text{width}} - 0.5 \right) \tag{3.1}$$

$$\phi = \pi \left(\frac{y}{\text{height}} - 0.5 \right) \tag{3.2}$$

where:

- θ is the **azimuthal angle** (longitude), spanning $[-\pi, \pi]$.
- ϕ is the **polar angle** (latitude), spanning $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$.

Next, we calculate the cartesian unit vectors using standard spherical-to-Cartesian transformations:

$$X_{\text{proj}} = \sin(\theta)\cos(\phi) \tag{3.3}$$

$$Y_{\text{proj}} = \sin(\phi) \tag{3.4}$$

$$Z_{\text{proj}} = \cos(\theta)\cos(\phi) \tag{3.5}$$

where:

- X_{proj} represents the horizontal direction.
- Y_{proj} represents the vertical direction.

- Z_{proj} represents the depth direction.

3.3 Custom U-Net Architecture

We use a custom U-Net [24] architecture in our pipeline to output the k-plane grids and density field grids. The entire set of layers can be found in the appendix (TODO: appendix)

The input to our U-Net is the image at the scene center: $(x, y, z) = (0, 0, 0)$. We start with initializing a ResNet101 [13] backbone as the downsampling layers. The architecture of ResNet101 can be referenced in Fig. 3.2. In ResNet101, conv3_1, conv4_1, and conv5_1 perform downsampling with a stride of 2.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 3.2: *Architecture for ResNet101. Building blocks are shown in brackets, with the number of blocks stacked. Reproduced from ResNet [13].*

Our upsampling blocks are custom-built layers built for two functions. First, it is a core feature of the U-Net architecture. These blocks upsample the inputs until they return to their original resolution. Second, in our method, they act as intermediaries that output parts of our k-planes field grid structure and the density field grid structure.

3.4 Scene Contraction

Our model contracts unbounded space using the contraction proposed in Mip-NeRF 360 [1]. In Mip-NeRF 360, Barron et al. explain that neural radiance field methods struggle when dealing with unbounded scenes. An unbounded scene is a scene where the camera can point in any direction and scene content can exist at any distance. Scenes from the HM3D dataset contain rooms of different sizes, so we warp the space into a fixed volume with Mip-NeRF 360's contraction formula:

$$f(x) = \begin{cases} x, & \text{if } \|x\| \leq 1 \\ \left(2 - \frac{1}{\|x\|}\right) \frac{x}{\|x\|}, & \text{if } \|x\| > 1 \end{cases} \tag{3.6}$$

We contract the space to a ball of radius 2.

3.5 Ray Sampler

Our method adaptively samples points along a ray using a proposal network. In our proposal network, we set our first-stage sampler as a piecewise sampler that divides samples into two regions: the first half is spaced uniformly, and the second half is spaced with inverse depth. Our piecewise function to sample points is defined as:

$$f(x) = \begin{cases} \frac{x}{2}, & \text{if } x < 1 \\ 1 - \frac{1}{2x}, & \text{otherwise} \end{cases} \tag{3.7}$$

Outside of the first batch of samples, we use a probability density function (PDF) sampler [20]. We receive an input of ray samples and generate position samples given a weight distribution on predicted densities. Weights are obtained through alpha

composition [20], which can be simplified to:

$$w_i = \alpha_i * T_i \tag{3.8}$$

where α is alpha (opacity) and T is transmittance.

3.6 Optimization

Loss functions guide the optimization process of our method. We minimize the summation of four types of losses: MSE loss, PlaneTV loss, histogram loss, and distortion loss.

3.6.1 Mean Squared Error (MSE) Loss

Our first reconstruction loss is MSE loss. MSE loss is the average squared difference, in pixels, between our rendered image and the ground truth image:

$$L_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 \tag{3.9}$$

3.6.2 Plane Total Variation (TV) Loss

In signal processing, total variation denoising is a noise removal process. It operates on the idea that signals with extra detail, or noise, have high total variation. Reducing the total variation loss removes noise while preserving important details such as edges. In our method, for each plane, the loss penalizes differences between neighboring pixels.

For the horizontal plane, total variation loss is:

$$TV_H = \sum_{b,c,h,w} (t_{b,c,h+1,w} - t_{b,c,h,w})^2 \tag{3.10}$$

where the sum is taken over all pixels except the last row.

For the vertical plane, total variation loss is:

$$TV_W = \sum_{b,c,h,w} (t_{b,c,h,w+1} - t_{b,c,h,w})^2 \tag{3.11}$$

where the sum is taken over all pixels except the last column.

For both, b is the batch size, c is the number of channels, h is the height of the plane, and w is the width of the plane.

Then, combining both horizontal and vertical variation losses gives us the final Plane Total Variation loss:

$$L_{TV} = 2\left(\frac{TV_H}{\text{count}_H} + \frac{TV_W}{\text{count}_W}\right) \tag{3.12}$$

where:

- $\text{count}_h = B \cdot C \cdot (H-1) \cdot W$: number of valid pixel pairs in height
- $\text{count}_w = B \cdot C \cdot H \cdot (W-1)$: number of valid pixel pairs in width

3.6.3 Histogram Loss

The histogram loss is the proposal loss as described in Mip-NeRF 360 [1]. This proposal loss is Barron et. al's method to supervise weights produced by the proposal network. This allows their larger NeRF model to be evaluated relatively few times,

and the smaller proposal network to be evaluated more times. This results in greatly improved rendering quality with only a modest increase in training time.

The loss function is given as

$$L_{\text{hist}}(\boldsymbol{t}, \boldsymbol{w}, \hat{\boldsymbol{t}}, \hat{\boldsymbol{w}}) = \sum_i \frac{1}{w_i} \max(0, w_i - \text{bound}(\hat{\boldsymbol{t}}, \hat{\boldsymbol{w}}, T_i))^2 \tag{3.13}$$

where t are normalized ray distances, w are weights, t' and w' are distances and weights from earlier proposal networks, and T is a histogram interval.

The function that computes the sum of all proposal weights that overlap with interval T is given as

$$\text{bound}(\hat{\boldsymbol{t}}, \hat{\boldsymbol{w}}, T) = \sum_{j:T \cap \hat{T}_j \neq \emptyset} \hat{w}_j \tag{3.14}$$

3.6.4 Distortion Loss

The distortion loss is another loss function proposed by Mip-NeRF 360 [1]. This loss function addresses "floaters" and "background collapse", both presented in Fig. 3.3, by encouraging volume rendering weights to be compact and sparse.

The loss function can be defined as

$$L_{\text{dist}}(\boldsymbol{t}, \boldsymbol{w}) = \sum_{i,j} w_i w_j \mid \frac{t_i + t_{i+1}}{2} - \frac{t_j + t_{j+1}}{2} \mid + \frac{1}{3} \sum_i w_i^2 (t_{i+1} - t_i) \tag{3.15}$$

where $\boldsymbol{t}$ are normalized ray distances, and $\boldsymbol{w}$ are weights. The first term minimizes the weighted distances between each pair of interval midpoints, and the second term minimizes the weighted length of each interval.

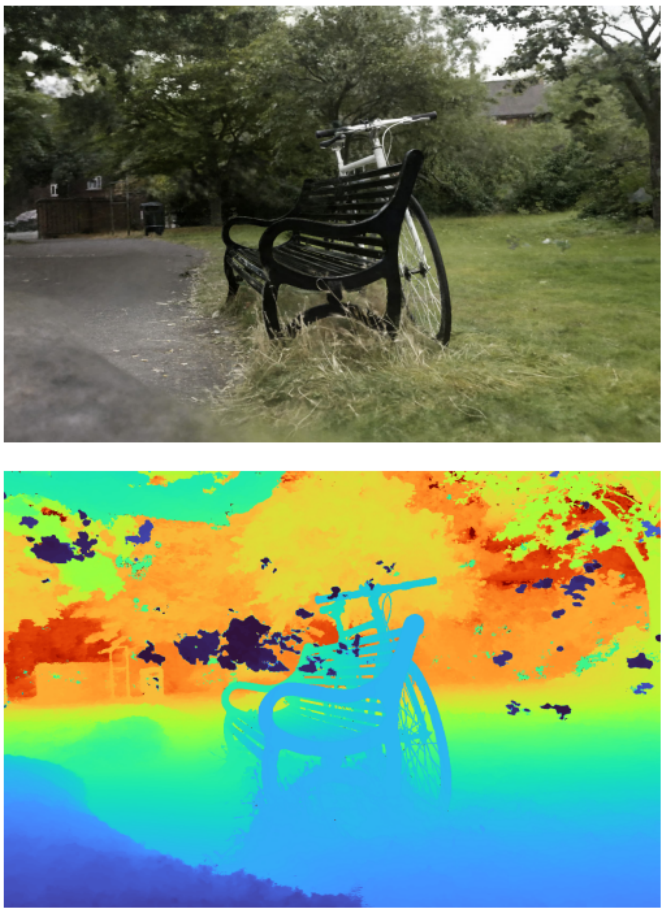


Figure 3.3: *"Floaters" are pieces of semi-transparent material floating in space. "Background collapse" is when the surfaces in the background collapse towards the camera (bottom left of depth map). Reproduced from [1].*

Thus, our final reconstruction loss is defined as:

$$L_{recon} = L_{\text{MSE}} + L_{TV} + L_{\text{hist}}(\boldsymbol{t}, \boldsymbol{w}, \hat{\boldsymbol{t}}) + L_{\text{dist}}(\boldsymbol{t}, \boldsymbol{w}) \tag{3.16}$$

Chapter 4

EXPERIMENTS

This chapter will describe our experimental setup to assess how well our method generates novel views from a set of 360° panorama images.

4.1 HM3D Dataset

We use the Habitat-Matterport 3D (HM3D) dataset [22] for our experiments. HM3D is a diverse, large-scale dataset of 1000 3D reconstructions from real-world locations. The scenes are represented as tectured 3D meshes, and feature residences, stores, and other indoor spaces. HM3D surpasses other photorealistic 3D datasets such as Replica [30] and MP3D [4] in higher visual fidelity, and is double the size of other building-scale datasets like MP3D.

Scene locations were scanned using a Matterport Pro2 tripod-based depth sensor. Scenes were verified by a group of 15 volunteer annotators who assessed the scenes based on scene quality. After an initial round of ratings, a group of three expert annotators finalized the scenes that would be part of the HM3D dataset.

4.2 Grid Structure

The shapes of our k-planes multiscale field grid and density field multiscale grid are shown in Table 4.1. A multiscale grid encourages spatial smoothness [11], so our method uses the spatial resolutions 128, 256, 512, and 1024 for the k-plane field grids, and the spatial resolutions 128 and 256 for the density field grids. The grids

have the shape M x N x N, where M is the size of the stored features that capture the color and density of the scene, and N is the spatial resolution.

Table 4.1: *Multiscale Plane Structure for K-Planes Field Grids and Density Field Grids*

Grid Type	Level 1	Level 2	Level 3	Level 4
K-Plane Field Grids	16 x 128 x 128	16 x 256 x 256	16 x 512 x 512	16 x 1024 x 1024
	16 x 128 x 128	16 x 256 x 256	16 x 512 x 512	16 x 1024 x 1024
	16 x 128 x 128	16 x 256 x 256	16 x 512 x 512	16 x 1024 x 1024
Density Field Grids	8 x 128 x 128	8 x 256 x 256	-	-
	8 x 128 x 128	8 x 256 x 256	-	-
	8 x 128 x 128	8 x 256 x 256	-	-

4.3 Experimental Setup

Most scenes from the HM3D dataset contain 25 different viewpoints, rendered as a 5x5 grid where each pose is placed 20cm apart. The 25 poses are shown in Table 4.2.

From these viewpoints, we select only a few of them for our experiments using a limited number of training images. We select the center viewpoint (012) to be our U-Net training image throughout all experiments. Table 4.3 shows the poses used in our experiment.

For training, we use the Adam optimizer with a starting learning rate of 0.01. For our reconstruction loss, we use a PlaneTV loss weight of 0.0001, a histogram loss weight of 1.0, and a distortion loss weight of 0.001. We train our model for 5,000 iterations for each scene on a single NVIDIA Tesla V100 GPU, which takes about 50 minutes to run.

Table 4.2: *Poses*

Name	X_In	Y_Right	Z_Up
000	0.0	0.4	0.4
001	0.0	0.2	0.4
002	0.0	0.0	0.4
003	0.0	-0.2	0.4
004	0.0	-0.4	0.4
005	0.0	0.4	0.2
006	0.0	0.2	0.2
007	0.0	0.0	0.2
008	0.0	-0.2	0.2
009	0.0	-0.4	0.2
010	0.0	0.4	0.0
011	0.0	0.2	0.0
012	0.0	0.0	0.0
013	0.0	-0.2	0.0
014	0.0	-0.4	0.0
015	0.0	0.4	-0.2
016	0.0	0.2	-0.2
017	0.0	0.0	-0.2
018	0.0	-0.2	-0.2
019	0.0	-0.4	-0.2
020	0.0	0.4	-0.4
021	0.0	0.2	-0.4
022	0.0	0.0	-0.4
023	0.0	-0.2	-0.4
024	0.0	-0.4	-0.4

Table 4.3: *Training Viewpoints and Corresponding Poses. Only pose 12 (scene center) is used as the U-Net input.*

Number of Viewpoints	Poses Used
3-Viewpoint Experiment	0, 12 (scene center), 17, 24

4.4 Evaluation Metrics

We evaluate novel viewpoints rendered with our method by comparing them with ground truth viewpoints using the following image similarity metrics: PSNR, SSIM, and LPIPS.

4.4.1 Peak Signal-to-Noise Ratio

PSNR is a quality measurement, in decibels, between two images. The PSNR value can be interpreted as follows:

- Higher PSNR value: better reconstruction quality (less distortion)
- Lower PSNR value: lower reconstruction quality (more distortion)

PSNR is defined as:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{L}^2}{\text{MSE}} \right) \tag{4.1}$$

where L is the maximum possible intensity level (0-255 for an 8-bit image). MSE is the mean squared error and it is defined as:

$$\text{MSE} = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left(I(i,j) - K(i,j) \right)^2 \tag{4.2}$$

where I represents the matrix values of the ground truth image, and J represents the matrix values of the rendered image.

4.4.2 Structural Similarity Index Measure

SSIM is also an image quality metric that is more perceptually accurate. While PSNR compares just pixel values between images, SSIM compares the similarities between properties such as luminance, contrast, and structure. SSIM ranges between -1 and 1, where a SSIM value of 1 means the images are identical. SSIM is defined as:

$$\text{SSIM}(I,K) = \frac{(2\mu_I\mu_K + C_1)(2\sigma_{IK} + C_2)}{(\mu_I^2 + \mu_K^2 + C_1)(\sigma_I^2 + \sigma_K^2 + C_2)} \tag{4.3}$$

where:

- μ_I and μ_K are the mean intensities of images I and K.
- σ_I^2 and σ_K^2 are the variances of I and K.
- σ_{IK} is the covariance between I and K.
- C_1 and C_2 are small constants to prevent divison by zero.

4.4.3 Learned Perceptual Image Patch Similarity

LPIPS [33] is a deep-learning-based metric that measures the perceptual similarity between two images. While PSNR and SSIM use pixel values to calculate the metric, LPIPS uses internal activations from networks trained on high-level classification tasks that correspond to human perceptual judgment. LPIPS compares the L2 distance between feature maps extracted from VGGNet. The LPIPS value can be interpreted as follows:

- Lower LPIPS value: more perceptually similar
- Higher LPIPS value: more perceptual differences

Chapter 5

RESULTS

This chapter will describe the results from our experiments. We will show that our method on average, both quantitatively and qualitatively, outperforms K-Planes. All evaluation is done on scenes generated from the HM3D dataset.

Evaluation is performed on 141 scenes. A full table of metrics on every single evaluation scene can be found in the appendix.

5.1 Quantitative Evaluation

We begin by showing the average PSNR, SSIM, and LPIPS values across all 141 evaluation scenes. The quantitative results in Tab. 5.1 show that on average, our method outperforms K-Planes in PSNR by 1.69, while K-Planes achieves a slightly higher SSIM, beating our method by 0.01. SSIM compares small patches of an image rather than treating each pixel independently, so well-preserved local textures and details will lead to a high SSIM score. K-Planes also achieves a better average LPIPS score, beating our method by 0.05, likely for the same reasons. In section 5.2, we will show qualitative examples that illustrate the differences between the rendering quality for both methods and demonstrate that DeepPanoRF consistently produces higher-quality reconstructions, despite the differences in SSIM and LPIPS.

Fig. 5.1 shows a full comparison of the PSNR values for every evaluation scene.

For the PSNR metric, our method performs better in 79 scenes, and performs worse in 33 scenes. Table 5.2 shows the distribution of PSNR differences between our

Table 5.1: *Average PSNR, SSIM, and LPIPS for K-Planes and DeepPanoRF across all 141 evaluation scenes.*

Method	PSNR↑	SSIM↑	LPIPS↓
K-Planes	20.879	0.728	0.398
U-Net (Ours)	22.987	0.727	0.462

method and K-Planes across various ranges. Negative differences indicate that our approach performs better, and positive differences indicate that K-Planes performs better. The **[-10.205, -5.0]** range indicates scenes where our method performs considerably better than K-Planes. The **(-5.0, -3.0]** and **(-3.0, -1.0]** range indicate moderate improvements, and the **(-1.0, 0.0]** range indicates marginal improvements. Most of the scenes where our method performs better than K-Planes fall in the range [-1.0, -10.205], demonstrating a clear improvement in rendering quality.

Table 5.2: *PSNR difference ranges and counts. A negative PSNR value means [our method] performed better.*

PSNR Difference Ranges	Count
(-11.477, -5.0]	31
(-5.0, -3.0]	30
(-3.0, -1.0]	34
(-1.0, 0.0]	6
(0.0, 1.0]	10
(1.0, 3.0]	20
(3.0, 5.0]	6
(5.0, 10.655]	4

Fig. 5.2 shows a full comparison of the SSIM values for every evaluation scene, and Fig. 5.3 shows a full comparison of the LPIPS values for every evaluation scene.

For the SSIM and LPIPS metrics, our method is quantitatively outperformed by K-Planes. Table 5.3 and Table 5.4 shows the distribution of SSIM and LPIPS differences between our method and K-Planes across various ranges. For both, negative differences indicate a stronger performance from our method, and positive differences

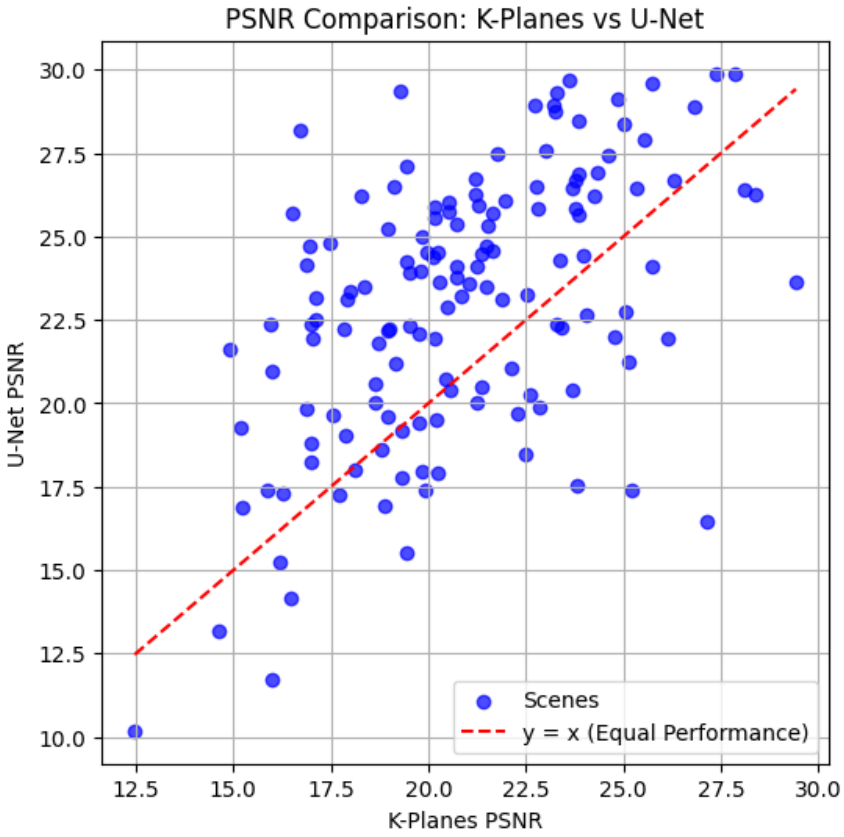


Figure 5.1: *K-Planes vs U-Net PSNR scatterplot. Each point represents a scene and the y=x line represents equal performance. Points ABOVE the line correspond to scenes where our U-Net method achieves a higher PSNR than K-Planes, while points BELOW the line correspond to scenes where K-Planes performs better.*

indicate that K-Planes performs better. For SSIM, most scenes perform similarly, with only a few scenes performing significantly better than the other. However, for LPIPS, there are a larger number of scenes where, quantitatively, K-Planes performs significantly better. In our qualitative evaluation, we will show that the LPIPS and SSIM metrics, in this case, are not accurate measures of how well a scene performs. Instead, PSNR is a better measure of performance.

Table 5.3: *SSIM difference ranges and counts. A negative SSIM difference indicates that [our method] performed better.*

SSIM Difference Ranges	Count
(-0.199, -0.1]	12
(-0.1, -0.05]	25
(-0.05, -0.01]	33
(-0.01, 0.0]	8
(0.0, 0.01]	4
(0.01, 0.05]	31
(0.05, 0.1]	14
(0.1, 0.287]	14

Table 5.4: *LPIPS difference ranges and counts. A negative LPIPS difference indicates that our method performed better.*

LPIPS Difference Ranges	Count
(-0.186, -0.1]	5
(-0.1, -0.05]	9
(-0.05, -0.01]	15
(-0.01, 0.0]	6
(0.0, 0.01]	4
(0.01, 0.05]	28
(0.05, 0.1]	21
(0.1, 0.361]	53

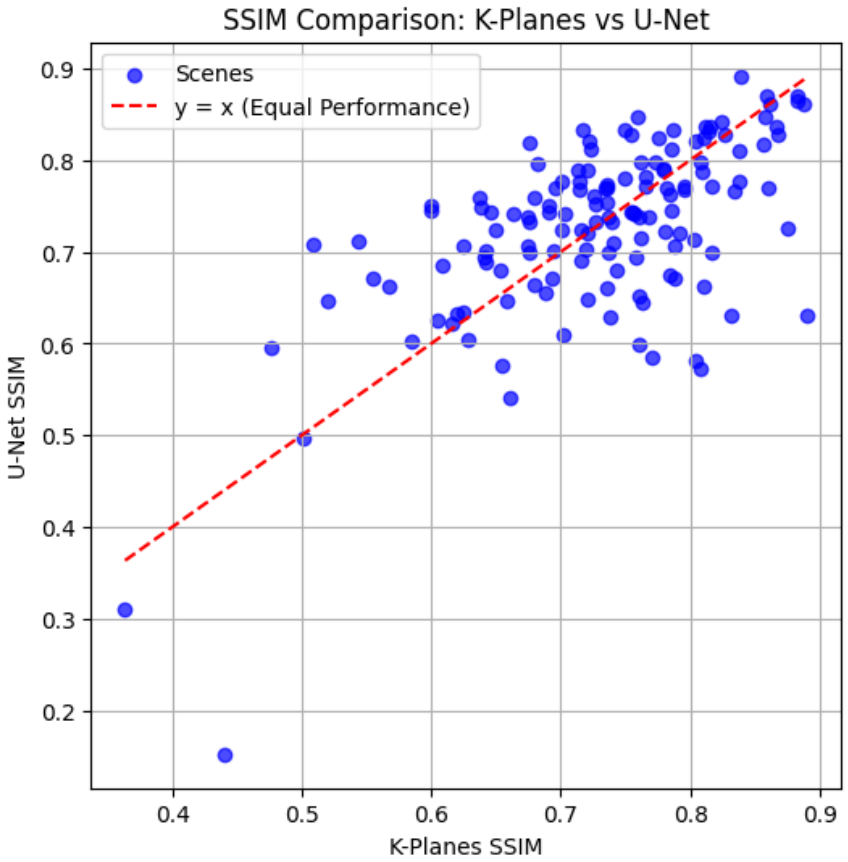


Figure 5.2: *K-Planes vs U-Net SSIM scatterplot. Each point represents a scene and the y=x line represents equal performance. Points ABOVE the line correspond to scenes where our U-Net method achieves a higher SSIM than K-Planes, while points BELOW the line correspond to scenes where K-Planes performs better.*

5.2 Qualitative Evaluation

The quantitative evaluation indicates that DeepPanoRF outperforms K-Planes in PSNR, and slightly underperforms K-Planes in SSIM. Qualitatively, however, our method, visually, performs better than K-Planes.

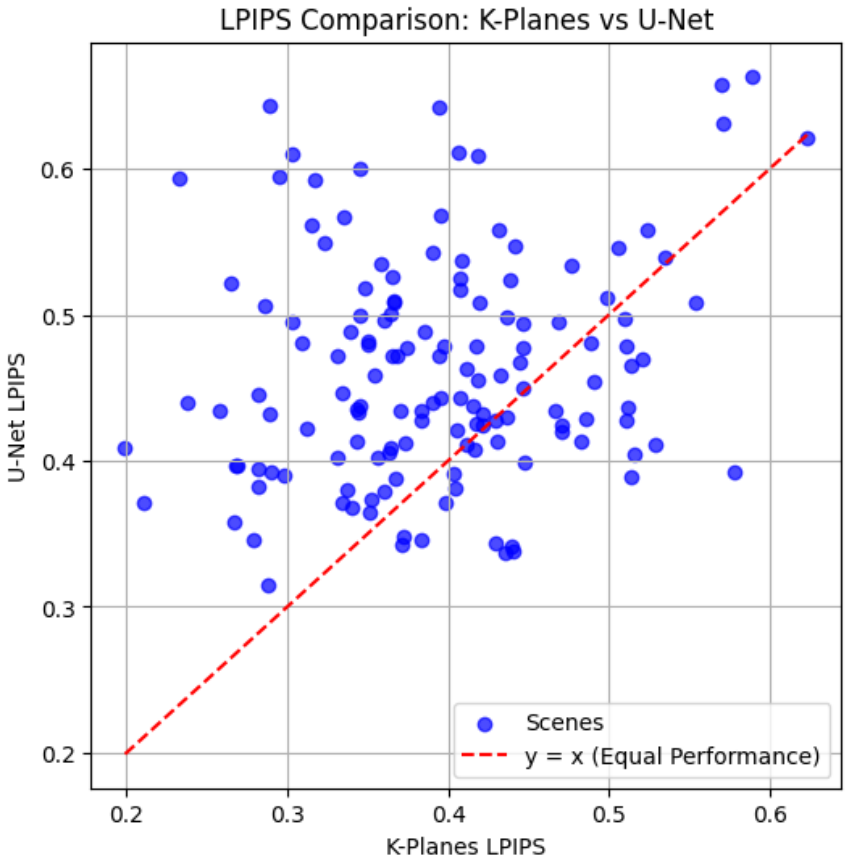


Figure 5.3: *K-Planes vs U-Net LPIPS scatterplot. Each point represents a scene and the y=x line represents equal performance. Points BELOW the line correspond to scenes where our U-Net method achieves a higher LPIPS than K-Planes, while points ABOVE the line correspond to scenes where K-Planes performs better.*

5.2.1 Simple scene example

Figure 5.4 shows the novel view renderings of scene "00400_000" with our method and K-Planes. In this scene, it is evident that K-Planes struggles to learn the scene representation with just three viewpoints, while our U-Net was successful in doing so. Our method outperforms K-Planes in PSNR 26.56 to 20.56, SSIM 0.794 to 0.737, and LPIPS 0.410 to 0.411. The K-Planes' rendering contains 'floater' artifacts throughout the entire scene, while ours only contains artifacts in localized areas, such as in place of the lamp.

Figure 5.4: *Qualitative comparison for scene '00400_000'.*

5.2.2 Complex scene example

Figure 5.5 shows the novel view renderings of a more difficult scene with more clutter: "00403_001". While both methods could render the scene's center decently, both struggled with the sides of the scene. Quantitatively, K-Planes outperformed our method in all three metrics: PSNR 19.42 to 18.16, SSIM 0.709 to 0.708, and LPIPS 0.344 to 0.449. Quantatively, K-Planes beat our method significantly in the LPIPS metric. However, we claim that our renderings achieved superior visual quality. Both approaches maintained decent reconstruction fidelity in the center but performed poorly towards the edges. Notably, our method's reconstruction generated fewer 'floater' artifacts, and was able to better capture the overall detail of the scene. For example, K-Planes struggled to understand the left and right hand sides of the scene, making the buckets in the scene almost impossible to recognize. However, while our

method lacks fine detail, the objects in the scene are identifiable as buckets, which shows a stronger grasp of the scene's overall composition.

Figure 5.5: *Qualitative comparison for scene '00403_001'.*

5.2.3 LPIPS metric example

Figure 5.6 shows the novel view renderings of scene "00420_003". While our method outperformed K-Planes in PSNR (26.25 vs. 23.75), K-Planes had better scores with SSIM and LPIPS. K-Planes outperformed our method in SSIM 0.782 to 0.776, and LPIPS 0.331 to 0.465. In the figure, we can see that our method's rendering appears blurrier and has less detail with the carpet, but contains fewer artifacts in the scene's center. However, K-Planes was unable to reconstruct the scene center, likely due to the limited detail with the sparse training images. The higher SSIM and LPIPS scores for K-Planes can be attributed to the few areas with a high-quality reconstruction,

while our method achieved a much higher PSNR score by better generalizing the scene from sparse input viewpoints.

Figure 5.6: *Qualitative comparison for scene '00420_003'.*

5.2.4 Error Map Comparison

Figures 5.7, 5.8, 5.9 illustrate impact of sparse-view reconstruction on both methods. The error maps visualize pixel-wise differences between the reconstructed rendering and ground truth, where brighter (white) regions correspond to higher error.

Overall, DeepPanoRF has significantly fewer high-error regions compared to K-Planes, and notably has a smaller number of "floaters". This suggests that DeepPanoRF can better understand the positioning of scene geometry under sparse conditions, leading to cleaner and more accurate reconstructions.

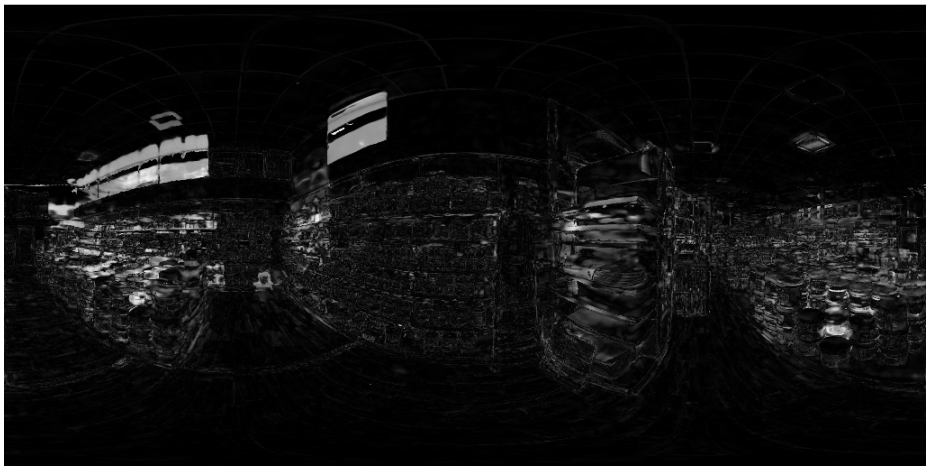
(a) *DeepPanoRF*

(b) *K-Planes*

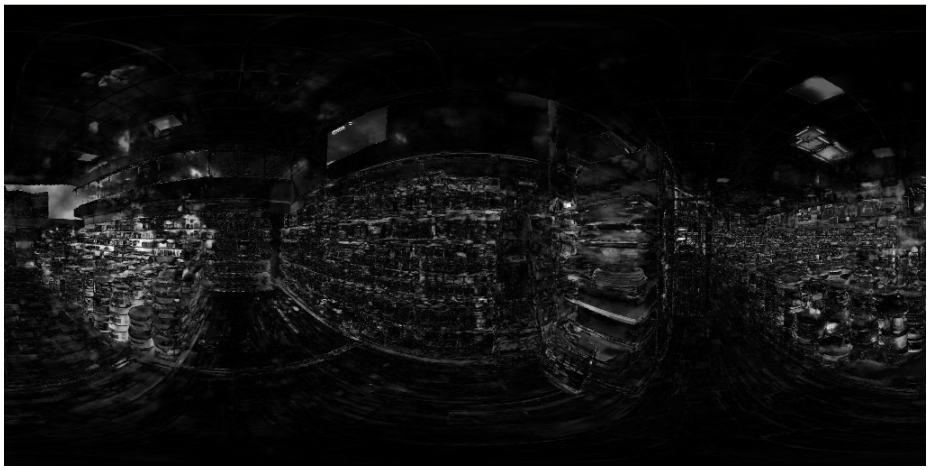
Figure 5.7: *Comparison of depth results between DeepPanoRF (top) and K-Planes (bottom) for scene '00400_000'.*

5.3 Discussion

K-Planes struggles with sparse-view scene reconstruction. The limited scene information from only a few input panoramas led to artifacts throughout the entire renderings, with only a few less affected areas. The ubiquitous 'floater' artifacts, which are objects being misplaced in 3D space due to the lack of information, suggest poor performance with sparse views. While K-Planes achieved a higher average SSIM score



(a) *DeepPanoRF*



(b) *K-Planes*

Figure 5.8: *Comparison of depth results between DeepPanoRF (top) and K-Planes (bottom) for scene '00403_001'.*

and a better LPIPS score, this was primarily due to localized regions of high-fidelity reconstruction, despite the overall rendering quality (PSNR) being poorer.

In contrast, DeepPanoRF excels on scenes with less clutter and still captures the overall structure, even in more complex scene environments. However, it struggles to reconstruct the fine details of scenes, leading to lower SSIM and LPIPS scores. We believe that this is because our U-Net backbone is a ResNet101. A more powerful backbone like a ResNet152 or a more powerful transformer-based model might help with the fine details.

(a) *DeepPanoRF*

(b) *K-Planes*

Figure 5.9: *Comparison of depth results between DeepPanoRF (top) and K-Planes (bottom) for scene '00420_003'.*

Chapter 6

FUTURE WORK

This chapter will detail some limitations of DeepPanoRF and explore potential directions for future research. Our results show that a U-Net is effective at parameterizing feature grids for 360° panoramas to enable 360° scene reconstructions from a sparse input set, but there are several areas of potential improvement:

1. A single U-Net that can generalize across multiple scenes.
2. Extend evaluation to real-world data.
3. Integrate with other sparse-view methods

6.1 Generalizable U-Net

Our current pipeline utilizes a U-Net architecture to parameterize the feature grids, but it is restricted to single-scene training, meaning the same U-Net cannot be used to render viewpoints across multiple scenes. In practice, a model that is able to generalize across multiple scene environments without per-scene optimization would be the goal. This would be highly desirable in real-time applications such as VR and robotics.

One possible approach would be to use a more powerful deep-learning architecture, such as a transformer model. This would also help with learning the small details of a scene, which our model currently struggles with. Another approach would be to fine-tune a pre-trained model instead of training from scratch for every scene. Finally, we

could also explore using network architectures specifically designed for panoramas. Unlike standard images, a property of panoramas is horizontal wrapping, meaning the left and right edges of the image are connected in a continuous manner.

6.2 Real-world Data

DeepPanoRF performs all training and evaluation on the HM3D dataset, which consists only of synthetic, high-quality, indoor environments. The next logical extension would be to evaluate the model on real-world panoramic data, which often contains more variability in lighting, noise, and scene complexity.

Additionally, real-world data is often imperfectly captured, especially when taken by hand, leading to rotational misalignment between views. A possible extension would be to evaluate the effect of these misalignments and to explore methods that correct for them.

6.3 Sparse View Extension

To extend our model's capability in sparse-view scenarios, we draw inspiration from recent advances that explicitly address the limitations of neural rendering under sparse views.

6.3.1 Incorporating Flow Matching Models with FlowR

FlowR [10] proposes a novel pipeline that leverages a flow matching model to generate extra high-quality views. By modeling the velocity field between incorrect renderings and the real ground truth images of a specific viewpoint, FlowR can constrain its

generative model to existing content. We can incorporate something similar in DeepPanoRF: learn a flow to connect possibly sparse reconstructions to expected dense reconstructions.

6.3.2 Adversarial Supervision with GANeRF

GANeRF [23] introduces adversarial learning into the NeRF optimization process to compensate for a potential lack of precisely collected input data in in-the-wild use cases. GANeRF resolves imperfections directly in the NeRF reconstruction. Using an adversarial loss enforces the radiance field representation to produce image patches that more closely resemble the distribution of real-world image patches, which helps minimize quality issues that arise from sparse input data.

Chapter 7

CONCLUSION

We present DeepPanoRF, a novel method for learning 360° scene reconstructions from a sparse input set of just three viewpoint panoramas. DeepPanoRF builds upon K-Planes[11], a radiance field framework that stores explicit features on orthogonal feature planes. Instead of directly optimizing the feature grids, we use a custom U-Net architecture, without any priors, to jointly parameterize a k-planes feature grid and a density grid. We train and evaluate our method using the HM3D dataset, which contain high-quality scenes represented as textured 3D meshes and features residences, stores, and other indoor spaces. We find that DeepPanoRF surpasses K-Planes both quantitatively and qualitatively. Our method performs well on scenes with less clutter and can still capture the overall structure of more complex scenes. Additionally, our renderings have fewer artifacts when reconstructing from a sparse input set, which are prevalent in K-Planes' renderings.

Our main technical contributions are:

1. We create a pipeline (DeepPanoRF) that reconstructs 360° panoramas using scenes from the HM3D dataset.
2. We create a U-Net that parameterizes k-plane and density feature grids.
3. Our method, on average, outperforms K-Planes with sparse views.

DeepPanoRF is a step towards sparse-view 360° scene reconstruction. In real-world scenarios, acquiring dense views of a scene is not always practical. Due to cost or time constraints, capturing a large number of images for dense reconstruction can

be unsuitable for consumer applications and casual use. By enabling scenes to be reconstructed with just a few viewpoints, our approach can be useful in fields like robotics, virtual reality, and content creation.

Our method has a few limitations. First of all, our U-Net's backbone is a ResNet101, which may not be optimal for capturing the fine details of scenes. A more powerful backbone such as ResNet152 or a vision transformer (ViT) could improve the fidelity of the small details and improve performance in the more complex scenes that our method struggles with. Second, our model is trained and evaluated on the HM3D dataset, which consists solely of synthetic indoor environments. Our method may not generalize well if presented with outdoor scenes or real scenes.

As research in 3D scene reconstruction continues to evolve, new methods will push the boundaries of what is possible with sparse-view inputs. These future advancements will further refine view synthesis from limited viewpoints, improving both fidelity and efficiency. DeepPanoRF takes a step towards practical 360° scene reconstruction, and we hope it inspires further innovation in panoramic view synthesis, enabling richer and more immersive virtual environments.

BIBLIOGRAPHY

[1] J. T. Barron, B. Mildenhall, D. Verbin, et al. "Mip-nerf 360: Unbounded anti-aliased neural radiance fields". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 5470–5479.

[2] A. Blattmann, T. Dockhorn, S. Kulal, et al. "Stable video diffusion: Scaling latent video diffusion models to large datasets". In: *arXiv preprint arXiv:2311.15127* (2023).

[3] E. R. Chan, C. Z. Lin, M. A. Chan, et al. "Efficient geometry-aware 3d generative adversarial networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 16123–16133.

[4] A. Chang, A. Dai, T. Funkhouser, et al. "Matterport3d: Learning from rgb-d data in indoor environments". In: *arXiv preprint arXiv:1709.06158* (2017).

[5] A. Chen, Z. Xu, A. Geiger, et al. "Tensorf: Tensorial radiance fields". In: *European conference on computer vision*. Springer. 2022, pp. 333–350.

[6] Y. Chen, H. Xu, C. Zheng, et al. "Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images". In: *European Conference on Computer Vision*. Springer. 2024, pp. 370–386.

[7] Y. Chen, C. Zheng, H. Xu, et al. "Mvsplat360: Feed-forward 360 scene synthesis from sparse views". In: *arXiv preprint arXiv:2411.04924* (2024).

[8] Z. Chen, Y.-P. Cao, Y.-C. Guo, et al. "Panogrf: Generalizable spherical radiance fields for wide-baseline panoramas". In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 6961–6985.

[9] P. Debevec, Y. Yu, and G. Borshukov. "Efficient view-dependent image-based rendering with projective texture-mapping". In: *Rendering Techniques' 98: Proceedings of the Eurographics Workshop in Vienna, Austria, June 29—July 1, 1998 9*. Springer. 1998, pp. 105–116.

[10] T. Fischer, S. R. Bulò, Y.-H. Yang, et al. "FlowR: Flowing from Sparse to Dense 3D Reconstructions". In: *arXiv preprint arXiv:2504.01647* (2025).

[11] S. Fridovich-Keil, G. Meanti, F. R. Warburg, et al. "K-planes: Explicit radiance fields in space, time, and appearance". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 12479–12488.

[12] Y. Furukawa, C. Hernández, et al. "Multi-view stereo: A tutorial". In: *Foundations and trends® in Computer Graphics and Vision* 9.1-2 (2015), pp. 1–148.

[13] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[14] R. Heckel and M. Soltanolkotabi. "Denoising and regularization via exploiting the structural bias of convolutional generators". In: *arXiv preprint arXiv:1910.14634* (2019).

[15] A. Jain, M. Tancik, and P. Abbeel. "Putting nerf on a diet: Semantically consistent few-shot view synthesis". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5885–5894.

[16] M. Kim, S. Seo, and B. Han. "Infonerf: Ray entropy minimization for few-shot neural volume rendering". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 12912–12921.

[17] M. Levoy and P. Hanrahan. "Light field rendering". In: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023, pp. 441–452.

[18] R. Li, H. Gao, M. Tancik, and A. Kanazawa. "Nerfacc: Efficient sampling accelerates nerfs". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2023, pp. 18537–18546.

[19] L. McMillan and G. Bishop. "Plenoptic modeling: An image-based rendering system". In: *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 2023, pp. 433–440.

[20] B. Mildenhall, P. P. Srinivasan, M. Tancik, et al. "Nerf: Representing scenes as neural radiance fields for view synthesis". In: *Communications of the ACM* 65.1 (2021), pp. 99–106.

[21] J. J. Park, P. Florence, J. Straub, et al. "Deepsdf: Learning continuous signed distance functions for shape representation". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 165–174.

[22] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, et al. "Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai". In: *arXiv preprint arXiv:2109.08238* (2021).

[23] B. Roessle, N. Müller, L. Porzi, et al. "Ganerf: Leveraging discriminators to optimize neural radiance fields". In: *ACM Transactions on Graphics (TOG)* 42.6 (2023), pp. 1–14.

[24] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer. 2015, pp. 234–241.

[25] L. I. Rudin, S. Osher, and E. Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: nonlinear phenomena* 60.1-4 (1992), pp. 259–268.

[26] M. Savva, A. Kadian, O. Maksymets, et al. “Habitat: A platform for embodied ai research”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9339–9347.

[27] S. M. Seitz and C. R. Dyer. “Photorealistic scene reconstruction by voxel coloring”. In: *International journal of computer vision* 35 (1999), pp. 151–173.

[28] R. Shi, X. Wei, C. Wang, and H. Su. “Zerorf: Fast sparse view 360deg reconstruction with zero pretraining”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 21114–21124.

[29] V. Sitzmann, J. Thies, F. Heide, et al. “Deepvoxels: Learning persistent 3d feature embeddings”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2437–2446.

[30] J. Straub, T. Whelan, L. Ma, et al. “The replica dataset: A digital replica of indoor spaces”. In: *arXiv preprint arXiv:1906.05797* (2019).

[31] D. Ulyanov, A. Vedaldi, and V. Lempitsky. “Deep image prior”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9446–9454.

[32] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson. “Synsin: End-to-end view synthesis from a single image”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 7467–7477.

[33] R. Zhang, P. Isola, A. A. Efros, et al. “The unreasonable effectiveness of deep features as a perceptual metric”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 586–595.

Appendix A

APPENDIX A

Example scene renderings.

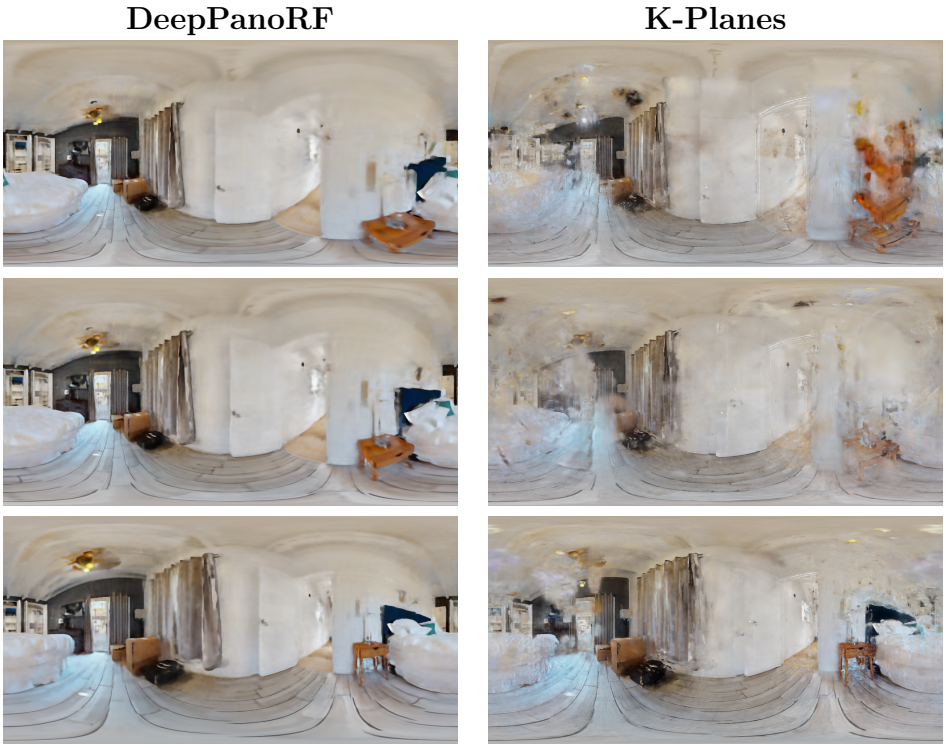


Figure A.1: *Scene 00400_000*

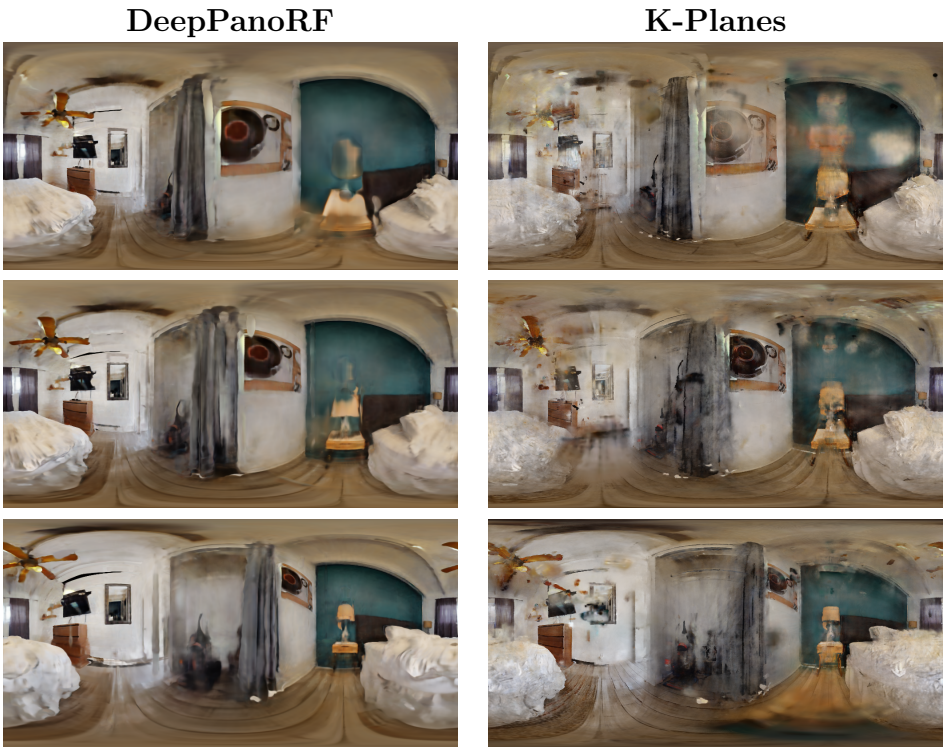


Figure A.2: *Scene 00400_001*

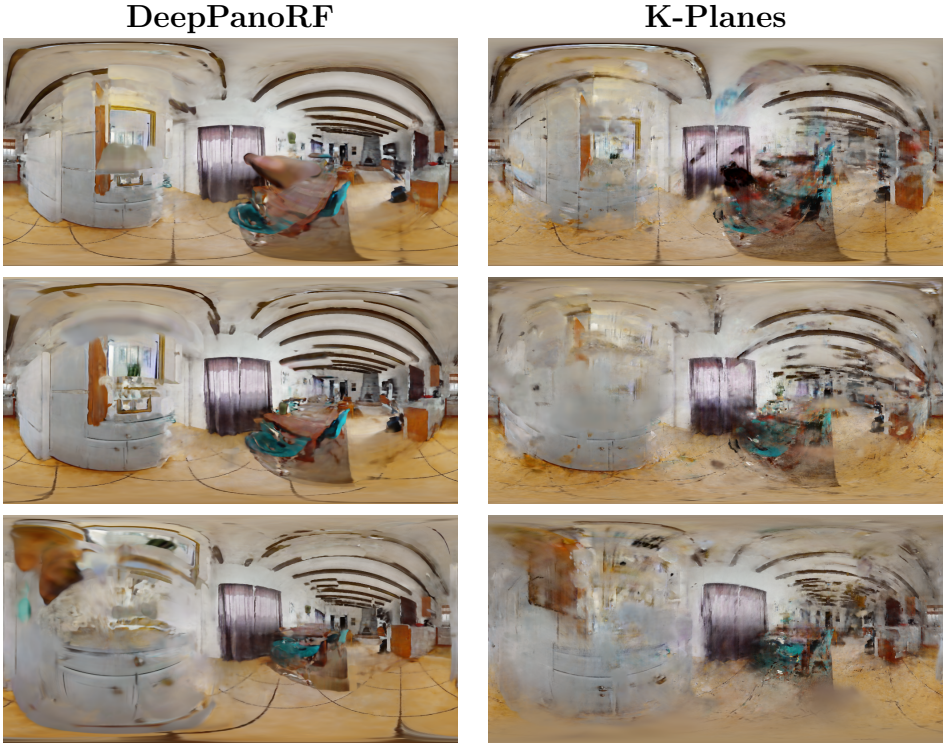


Figure A.3: *Scene 00400_002*

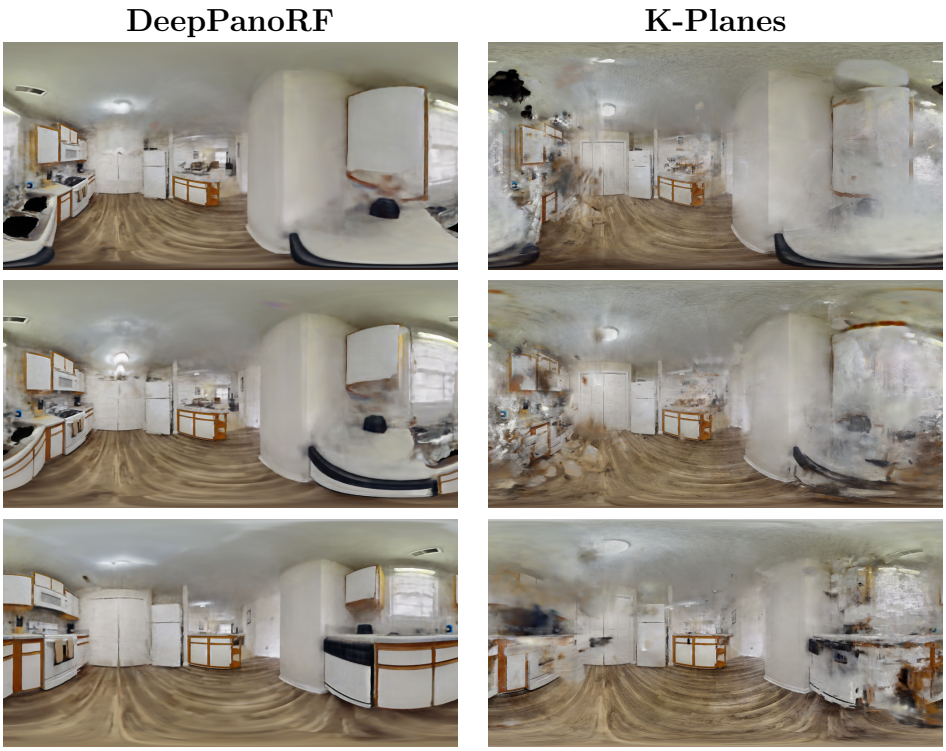


Figure A.4: *Scene 00401_000*

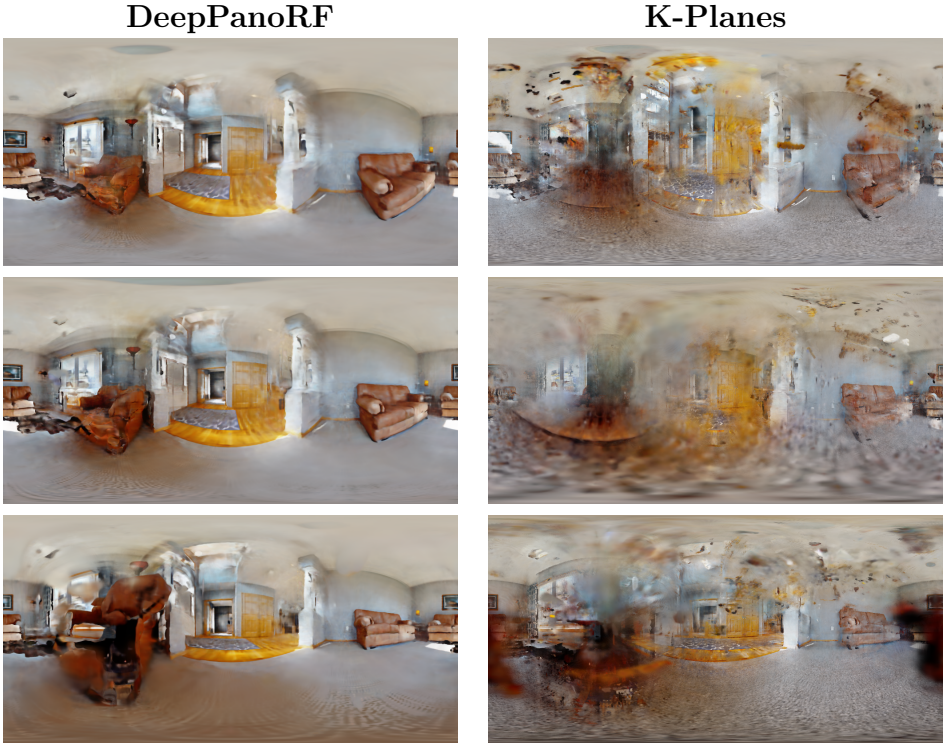


Figure A.5: *Scene 00405_000*

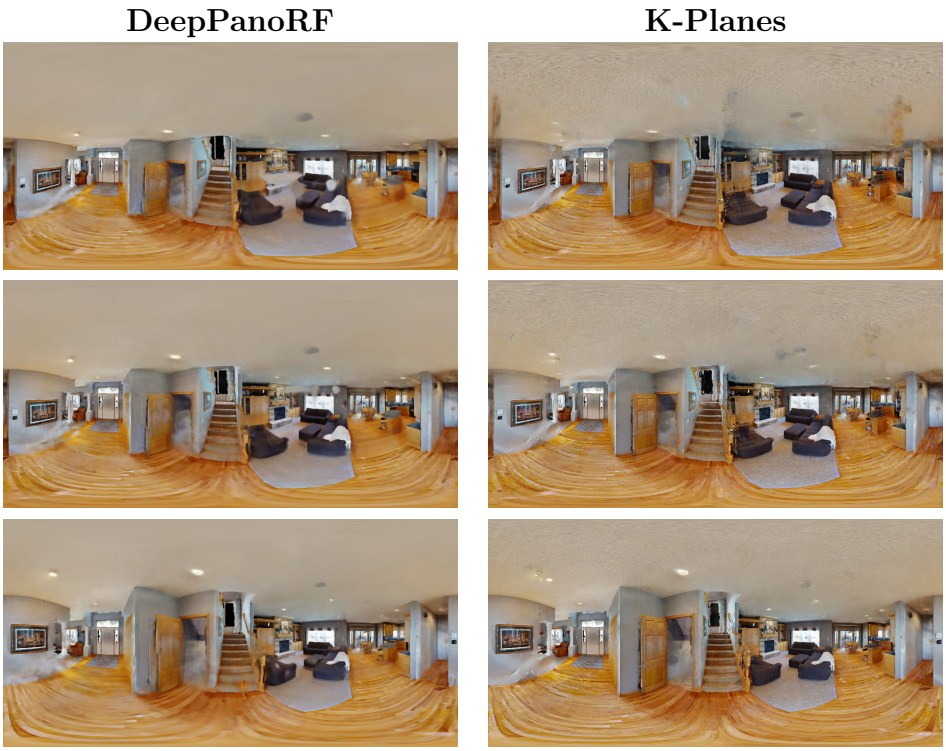


Figure A.6: *Scene 00405_001*

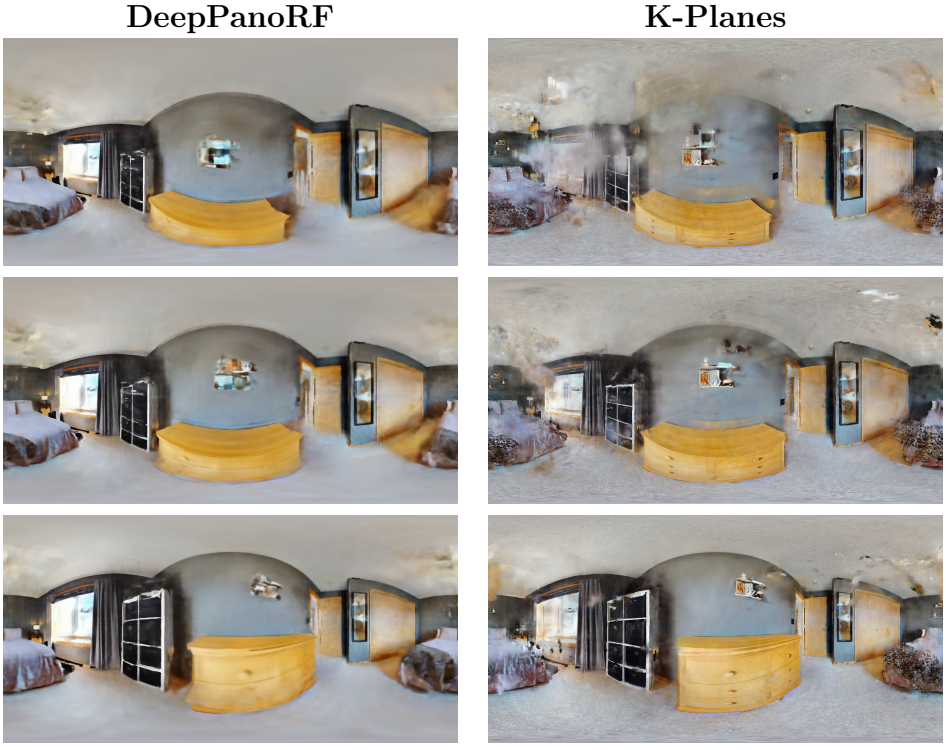


Figure A.7: *Scene 00405_002*

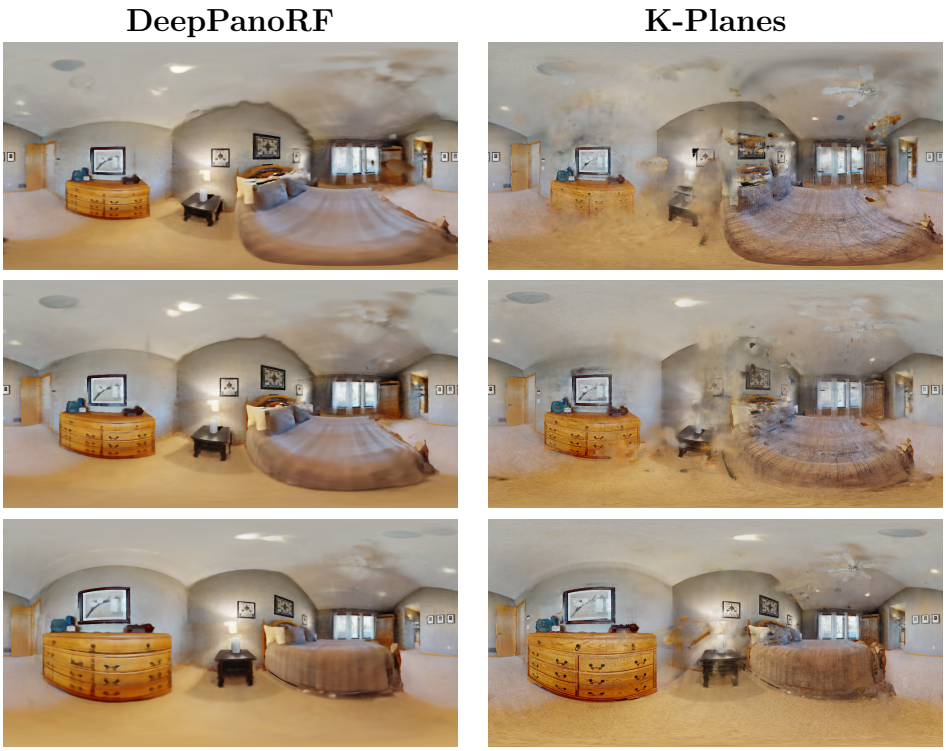


Figure A.8: *Scene 00405_003*

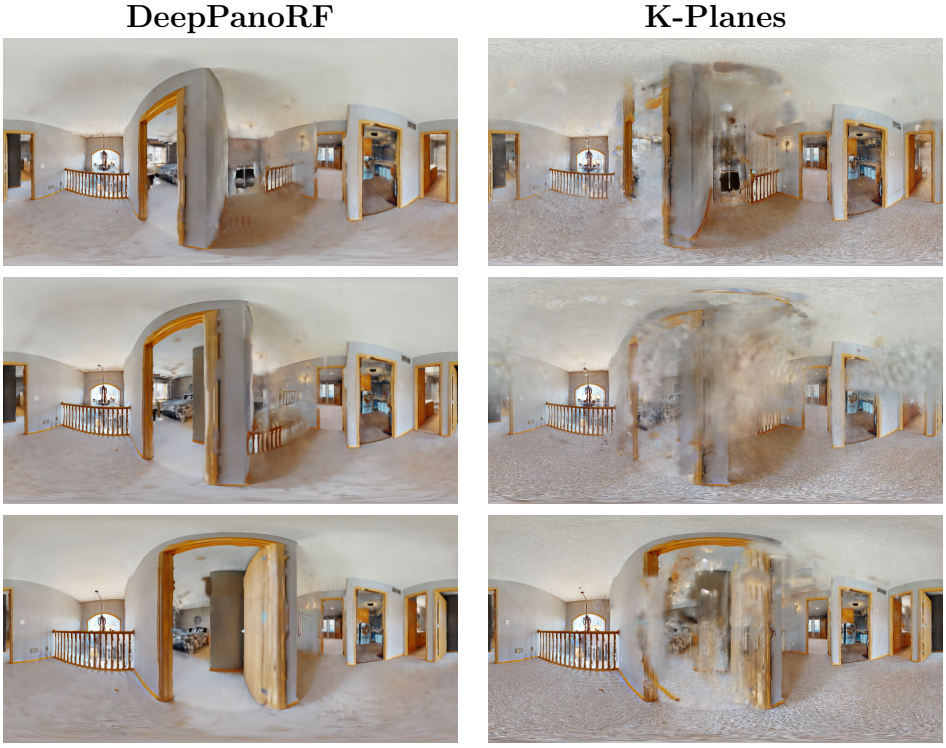


Figure A.9: *Scene 00405_004*

Appendix B

APPENDIX B

Table B.1: *Per-Scene Evaluation Metrics for DeepPanoRF and K-Planes*

	DeepPanoRF			**K-Planes**		
Scene	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
00400_000	27.09	0.812	0.411	19.43	0.724	0.411
00400_001	22.32	0.732	0.425	19.54	0.676	0.418
00400_002	18.24	0.663	0.436	16.98	0.568	0.511
00401_000	23.37	0.706	0.467	18.00	0.624	0.445
00401_002	22.01	0.672	0.549	24.76	0.788	0.323
00402_000	22.37	0.714	0.422	23.29	0.761	0.312
00402_003	24.83	0.751	0.338	17.46	0.599	0.435
00403_000	22.64	0.827	0.372	24.06	0.868	0.211
00403_001	18.61	0.724	0.433	18.81	0.700	0.344
00404_000	20.48	0.771	0.427	21.35	0.796	0.384
00404_001	22.24	0.769	0.481	18.99	0.735	0.489
00404_002	14.18	0.541	0.538	16.48	0.660	0.408
00404_004	19.70	0.766	0.535	22.28	0.834	0.358
00404_005	26.03	0.773	0.439	20.53	0.736	0.391
00404_006	22.28	0.713	0.495	23.42	0.803	0.303
00404_007	25.95	0.776	0.443	21.28	0.714	0.395
00405_000	21.94	0.595	0.508	17.03	0.476	0.554
00405_001	26.41	0.722	0.440	28.08	0.780	0.238
00405_002	23.22	0.626	0.543	20.84	0.605	0.391
00405_003	26.43	0.732	0.500	23.67	0.739	0.345

	DeepPanoRF			K-Planes		
Scene	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
00405_004	24.12	0.603	0.496	20.71	0.584	0.360
00406_002	22.18	0.744	0.420	18.97	0.691	0.471
00406_003	17.26	0.605	0.524	17.71	0.628	0.438
00406_004	18.46	0.585	0.562	22.48	0.771	0.315
00407_001	29.61	0.824	0.402	25.73	0.810	0.331
00407_003	19.90	0.663	0.566	22.86	0.810	0.335
00408_003	27.44	0.847	0.346	24.62	0.858	0.279
00409_001	29.70	0.797	0.459	23.62	0.773	0.354
00409_004	22.07	0.680	0.479	19.77	0.654	0.418
00409_006	23.13	0.738	0.495	17.91	0.675	0.469
00410_000	26.75	0.825	0.458	21.20	0.776	0.433
00410_002	23.14	0.737	0.517	21.87	0.760	0.408
00410_003	26.22	0.768	0.437	24.24	0.795	0.346
00411_000	26.43	0.861	0.392	25.35	0.888	0.290
00411_002	29.37	0.848	0.399	19.29	0.759	0.447
00411_003	25.71	0.796	0.343	16.52	0.682	0.429
00411_004	22.49	0.712	0.411	17.12	0.544	0.529
00412_000	27.91	0.787	0.391	25.54	0.809	0.298
00412_001	23.28	0.680	0.518	22.52	0.743	0.348
00412_002	27.46	0.769	0.406	21.77	0.736	0.364
00412_003	28.94	0.836	0.374	22.73	0.815	0.352
00412_004	19.42	0.703	0.409	19.76	0.720	0.364
00413_001	29.89	0.865	0.481	27.87	0.883	0.309
00413_002	29.29	0.891	0.381	23.28	0.839	0.404
00414_002	26.26	0.769	0.435	28.39	0.860	0.258
00414_005	19.53	0.755	0.472	20.19	0.735	0.394

	DeepPanoRF			K-Planes		
Scene	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
00415_000	23.16	0.742	0.497	17.13	0.645	0.510
00415_001	19.04	0.702	0.546	17.87	0.694	0.506
00415_002	23.51	0.723	0.424	18.37	0.649	0.471
00415_004	21.04	0.720	0.508	22.14	0.792	0.366
00416_000	23.77	0.768	0.428	20.74	0.715	0.430
00416_001	17.29	0.646	0.558	16.26	0.658	0.524
00416_002	23.65	0.781	0.407	20.29	0.750	0.416
00416_003	17.40	0.684	0.469	15.87	0.608	0.521
00416_004	21.93	0.664	0.488	20.17	0.680	0.386
00417_000	29.12	0.860	0.397	24.87	0.861	0.268
00417_002	24.08	0.837	0.413	21.24	0.811	0.431
00418_002	18.01	0.649	0.508	18.12	0.720	0.420
00418_003	24.48	0.790	0.388	21.36	0.779	0.368
00418_004	26.94	0.742	0.471	24.31	0.756	0.369
00418_005	26.06	0.828	0.412	21.99	0.826	0.373
00418_006	29.88	0.871	0.358	27.37	0.883	0.267
00418_007	24.98	0.790	0.434	19.86	0.720	0.467
00418_008	28.96	0.870	0.372	23.21	0.859	0.335
00418_009	25.64	0.763	0.435	23.87	0.785	0.344
00419_000	20.60	0.707	0.432	18.62	0.675	0.421
00419_001	17.54	0.582	0.610	23.80	0.804	0.303
00419_002	24.72	0.820	0.368	21.49	0.804	0.340
00419_003	13.19	0.497	0.631	14.64	0.502	0.571
00420_000	20.03	0.672	0.547	18.65	0.693	0.441
00420_001	25.72	0.797	0.435	21.66	0.762	0.370
00420_002	23.51	0.724	0.437	21.48	0.716	0.415

	DeepPanoRF			K-Planes		
Scene	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
00420_003	25.85	0.769	0.472	23.75	0.782	0.331
00421_001	26.20	0.744	0.339	18.27	0.600	0.440
00421_002	25.73	0.762	0.430	20.53	0.726	0.437
00421_003	20.94	0.646	0.428	16.00	0.520	0.511
00421_005	25.35	0.742	0.348	20.72	0.664	0.372
00422_000	20.38	0.630	0.510	20.58	0.739	0.366
00422_001	19.61	0.732	0.450	18.94	0.727	0.447
00422_004	24.45	0.745	0.446	23.98	0.786	0.335
00423_000	20.72	0.699	0.478	20.43	0.737	0.374
00423_001	26.25	0.833	0.342	21.21	0.750	0.371
00423_002	26.68	0.809	0.397	26.28	0.838	0.269
00423_003	16.47	0.630	0.409	27.13	0.890	0.199
00423_004	17.79	0.660	0.481	19.33	0.735	0.350
00423_005	17.89	0.693	0.413	20.24	0.758	0.344
00423_006	23.58	0.812	0.434	21.05	0.786	0.384
00424_000	23.92	0.760	0.494	19.53	0.680	0.446
00424_001	23.63	0.726	0.594	29.42	0.875	0.233
00424_002	21.95	0.700	0.595	26.14	0.817	0.295
00425_001	21.64	0.708	0.392	14.93	0.509	0.578
00425_003	18.82	0.738	0.540	17.01	0.736	0.534
00425_004	25.32	0.789	0.381	21.51	0.779	0.337
00426_001	25.25	0.820	0.341	18.96	0.722	0.439
00426_002	28.36	0.817	0.382	25.03	0.857	0.282
00426_003	22.38	0.749	0.429	17.01	0.639	0.486
00426_004	26.48	0.833	0.391	19.13	0.786	0.404
00427_000	16.92	0.691	0.525	18.88	0.715	0.407

	DeepPanoRF			K-Planes		
Scene	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
00427_001	24.56	0.710	0.480	21.65	0.741	0.350
00427_003	15.53	0.577	0.611	19.44	0.655	0.406
00427_004	22.89	0.744	0.443	20.50	0.754	0.407
00427_006	25.56	0.788	0.425	20.18	0.713	0.422
00427_007	25.87	0.770	0.346	20.18	0.696	0.383
00428_002	20.03	0.837	0.558	21.24	0.866	0.431
00428_004	26.51	0.832	0.479	22.76	0.814	0.397
00428_005	28.18	0.833	0.454	16.70	0.717	0.491
00429_000	22.75	0.652	0.506	25.07	0.761	0.286
00429_001	26.87	0.743	0.394	23.87	0.755	0.282
00430_000	24.11	0.674	0.593	25.73	0.784	0.317
00430_001	27.60	0.771	0.472	22.99	0.765	0.366
00430_002	28.90	0.777	0.445	26.83	0.838	0.282
00430_003	28.49	0.739	0.489	23.87	0.767	0.340
00430_004	22.21	0.701	0.512	17.84	0.642	0.499
00430_005	26.66	0.721	0.501	23.75	0.721	0.364
00430_006	16.88	0.635	0.621	15.22	0.624	0.623
00431_000	11.72	0.153	0.642	15.97	0.440	0.394
00431_001	17.39	0.609	0.609	19.92	0.701	0.418
00432_000	24.29	0.706	0.432	23.35	0.788	0.289
00432_001	19.84	0.655	0.534	16.86	0.688	0.477
00432_002	24.55	0.752	0.379	19.97	0.727	0.361
00432_004	25.83	0.797	0.315	22.83	0.808	0.288
00432_005	17.40	0.572	0.643	25.22	0.808	0.289
00432_006	24.13	0.828	0.413	16.86	0.754	0.483
00432_007	19.64	0.632	0.499	17.57	0.619	0.436

	DeepPanoRF			K-Planes		
Scene	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
00432_008	21.27	0.631	0.521	25.13	0.832	0.265
00432_009	21.81	0.700	0.456	18.72	0.676	0.419
00432_010	21.21	0.689	0.478	19.16	0.642	0.511
00432_011	24.24	0.693	0.478	19.45	0.641	0.446
00432_012	20.38	0.772	0.526	23.68	0.817	0.366
00433_001	23.97	0.776	0.372	19.78	0.701	0.398
00433_002	24.51	0.742	0.365	20.26	0.703	0.351
00433_003	17.94	0.645	0.568	19.83	0.763	0.396
00433_004	10.19	0.310	0.657	12.47	0.364	0.570
00434_000	28.75	0.842	0.402	23.24	0.824	0.357
00434_001	19.17	0.783	0.421	19.32	0.766	0.406
00434_002	19.27	0.759	0.405	15.18	0.637	0.516
00434_003	24.70	0.819	0.389	16.97	0.676	0.514
00435_000	24.38	0.750	0.463	20.11	0.691	0.411
00435_001	15.24	0.621	0.663	16.19	0.616	0.589
00435_002	22.37	0.672	0.465	15.93	0.555	0.514
00436_000	20.26	0.598	0.600	22.60	0.760	0.345